DYNAMIC MATRIX CONTROL

A THESIS SUBMITTED IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND INSTRUMENTATION ENGINEERING

BY

RASHMI RANJAN KAR (10607030)

ANGSHUMAN KALITA(10607019)

Under the Guidance of

Prof. TARUN KUMAR DAN

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA

National Institute of Technology

ROURKELA

CERTIFICATE

This is to certify that the thesis entitled , "**DYNAMIC MATRIX CONTROL**" , submitted by **Mr. RASHMI RANJAN KAR** and **Mr. ANGSHUMAN KALITA** in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in '**ELECTRONICS AND INSTRUMENTATION ENGINEERING**' at the **National Institute of Technology , Rourkela** (Deemed University) is an authentic work carried out by them under my supervision.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other university / institute for the award of any Degree or Diploma.

Date

Prof. Tarun Kumar Dan
Dept. of Electronics and Communication Engg.
National Institute of Technology
Rourkela-769008

ACKNOWLEDGEMENT

We take this opportunity as a privilege to thank all individuals without whose support and guidance we could not have completed our project in this stipulated period of time.

First and foremost we would like to express our deepest gratitude to our Project Supervisor **Prof. Tarun Kumar Dan**, Department of Electronics and Communication Engineering, for his invaluable support, guidance, motivation and encouragement throughout the period this work was carried out.

We would also like to thank all the Professors and members of the Department of Electronics and Communication Engineering for their generous help in various ways for the completion of the thesis. We also extend our thanks to our fellow students for their friendly co-operation.

RASHMI RANJAN KAR
10607030

ANGSHUMAN KALITA
10607019

Dept. of E.C.E.
N.I.T. ROURKELA

ABSTRACT

Dynamic Matrix Control (DMC) was the first Model Predictive Control (MPC) algorithm introduced in early 1980s. These are proven methods that give good performance and are able to operate for long periods without almost any significant intervention. Model predictive control is also the only technique that is able to consider model restrictions. Today, DMC is available in almost all commercial industrial distributed control systems and process simulation software packages.

This project thesis provides a brief overview of Dynamic Matrix Control which is the backbone of Model Predictive Control. A brief history of early industrial MPC applications is given followed by some of its industrial uses. Then some basic structure of model predictive control is discussed. Then follows the three main integral parts of any Model Predictive Control algorithm which are the process model, the cost function and the optimization technique. Various process models like state space model, step response model and impulse response model are discussed followed by cost functions. Quadratic and absolute value cost functions are explained. The receding horizon technique is then explained which simplifies the optimization of the process models. A brief idea about the DMC tuning is given. Finally the simulation outputs under the MATLAB window are provided for the sake of conformity with the theoretical approaches.

List of Figures

Contents

Chapter 1

INTRODUCTION

1.1 DMC: What it is

Dynamic Matrix Control or in short DMC is a control algorithm designed explicitly to predict the future response of a plant. This algorithm was first developed by Shell Oil engineers in late 1970's and was intended for its use in petroleum refineries. Now-a-days its applications are found in a wide variety of areas including chemicals, food processing, automotive, and aerospace applications.

1.2 How it works

It is a form of control algorithm in which the current control action is obtained by solving a finite horizon of open loop optimal control problem using the current state of the plant as the initial state. This process is repeatedly done for each sampling point. The optimization yields an optimal control sequence and the first control in this sequence is applied to the plant.

In DMC, the models which are used, determine the behavior of complex dynamical systems. These models compensate for the effect of nonlinearities present in the variables and the chasm caused by non coherent process devolution. Hence the models are used to predict the behavior of dependent variables or outputs of the modeled dynamical system with respect to changes in the process independent variables or inputs.

In most processes, independent variables are most often set points of regulatory controllers that govern valve movement (e.g. valve positioners with or without flow, temperature or pressure controller cascades), while dependent variables are most often constraints in the process (e.g. product purity, equipment safe operating limits). The model predictive controller make use of the models and current plant measurements to calculate future moves in the independent variables that will result in operation that honors all independent and dependent variable constraints. The model predictive

controller then sends this set of independent variable moves to the corresponding regulatory controller set points which get implemented in the process.

Despite the fact that most real processes are approximately linear within only a limited operating window, linear MPC approaches are used in the majority of applications with the feedback mechanism of the MPC compensating for prediction errors due to structural mismatch between the model and the process. In model predictive controllers that consist only of linear models, the superposition principle of linear algebra enables the effect of changes in multiple independent variables to be added together to predict the response of the dependent variables. This simplifies the control problem to a series of direct matrix algebra calculations that are fast and robust. Hence it is called Dynamic Matrix Control.

When linear models are not sufficiently accurate because of process nonlinearities, the process can be controlled with nonlinear MPC. Nonlinear MPC utilizes a nonlinear model directly in the control application. The nonlinear model may be in the form of an empirical data or a high fidelity model based on fundamentals such as mass, species, and energy balances. The nonlinear models are linearized to derive a Kalman filter or specify a model for linear MPC. The time derivatives may be set to zero (steady state) for applications of real-time optimization or data reconciliation. Alternatively, the nonlinear model may be used directly in nonlinear model predictive control and nonlinear estimation (e.g. moving horizon estimation). A reliable nonlinear model is a core component of simulation, estimation, and control applications.

1.3 Early Industrial MPC Application

The use of MPC algorithm stretches back to early 1970's. The first model predictive control algorithm was Model Predictive Heuristic Control (MPHC), Richalet *et al* (Richalet 1978) in 1976. In 1979 Cutler and Ramaker presented their version called Dynamic Matrix Control (DMC), where control outputs were computed applying so called receding horizon principle. The time horizon was moved one step ahead during every control cycle and the optimization problem was solved repeatedly during every control cycle. An attractive feature of these predictive algorithms is their ability to deal with explicit constraints on system variables. Constraints are neglected in the majority of other control algorithms.

1.4 Various forms of MPC

Robust MPC: It provides guaranteed feasibility and stability of the process on which it is applied.
Feedback MPC: It mitigates shrinkage of feasible region.
Pre-computed MPC: It is an offline optimization process. Parameters are solved using linear or quadratic programming.
Decentralized MPC: It gives very fast response. Mainly this is used in automation process.

Chapter 2

Model Predictive Control

2.1 Concept of MPC

The main motive of Model Predictive Control is to find the input signal that best corresponds to some criterion which predicts how the system will behave applying this signal. The problem is converted into a mathematical model at a given state. The feedback strategy is derived solving this problem at each sampling time and using only the current control horizon. This is called the Receding horizon technique. This process can be summarized into four steps:

1. At sample time t compute future outputs, i.e the prediction, $Y(t+i);\ i = 1, ..., Np$ as function of future control inputs $u(t+i);\ i = 0, ..., Np - 1$.
2. Minimize the objective function re to $u(t+i);\ i = 0, ..., Np - 1$.
3. Apply $u(t)$ on the system.
4. At the next sampling time make $t = t + 1$ and go to Step 1.

2.2 Primary components of MPC

Model Predictive Control method consists of three main components. Those are namely:

1. The process model
2. The cost function
3. The optimizer

The process model includes the information about the controlled process and it is used to predict the response of the process values according to manipulated control variables. There after minimization of cost function ensures that the error is reduced. In the last step different optimization techniques are applied and the output gives the input sequence for the next prediction horizon.

2.3 Basic Structure of MPC

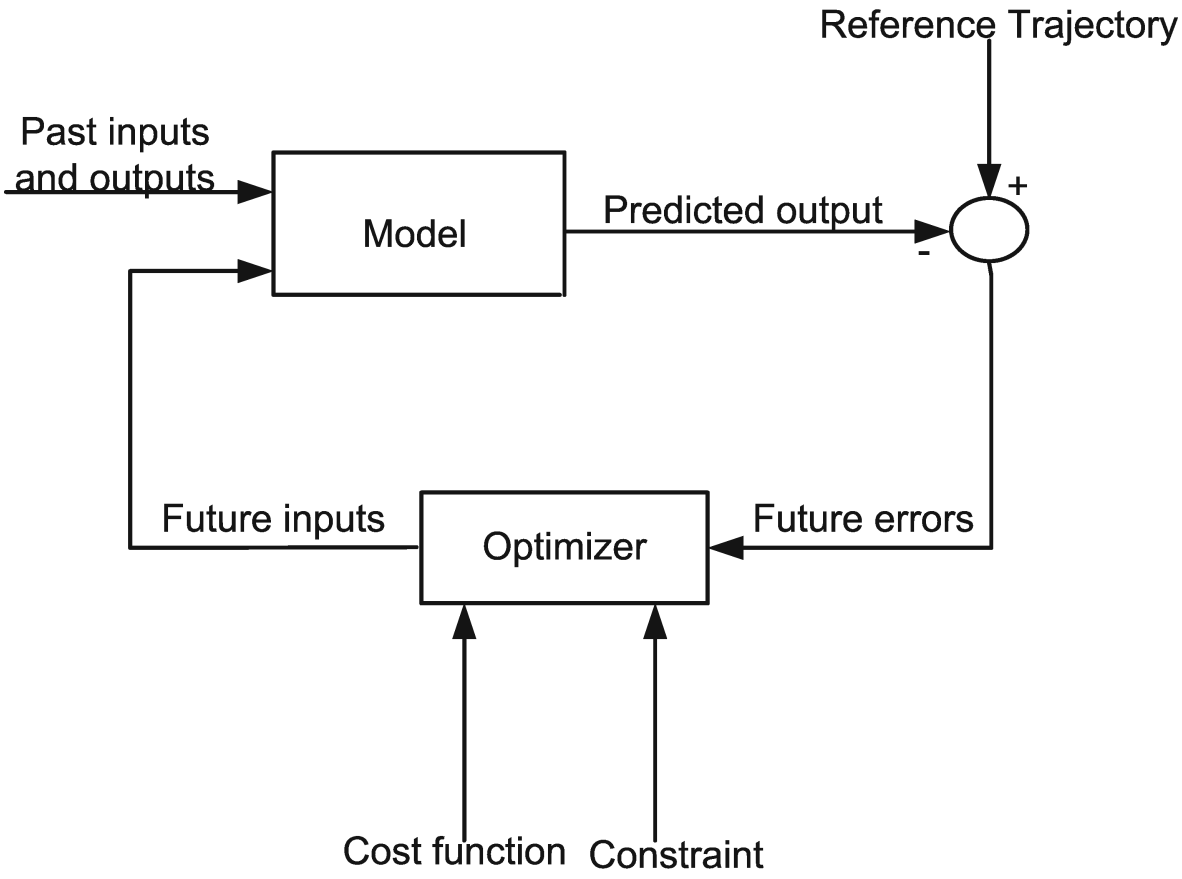

Figure 2.1: Basic structure of MPC (ref. [1])

2.4 Types of MPC

The MPC model can be broadly classified into two categories e.g. Linear MPC and Non-Linear MPC.

In case of Linear MPC, it uses linear model: $x(k+1) = Ax(k) + Bu(k)$

a quadratic cost function: $F = x'Qx + u'Ru$

linear constraints: $Hx(k) + Gu(k) < 0$

and a Quadratic program.

In case of Non-Linear MPC, it uses a uses nonlinear model: $x(k+1) = f(x, u)$

the cost function can be quadratic or non quadratic: $F(x,u)$

nonlinear constraints: $h(x,u) < 0$

and a Nonlinear program.

2.5 What makes MPC successful

There are various reasons for MPC to be so much successful. Some of the major contributing points are:

1. It handles structural changes.
2. It is an easy to tune method.
3. It allows operation closer to constraints, hence increased profit.
4. It can take account of actuator limitations.
5. It has plenty of time for on-line computations.
6. It can handle non-minimal phase and unstable processes.
7. It handles multivariable control problems naturally.

2.6 MPC Characteristics

1. It incorporates constraint values.
2. In MPC logic, an explicit system model is defined and used to predict future plant dynamics.
3. The receding horizon technique or moving horizon is implemented.
4. Performance oriented time domain formulation is there in MPC method.

2.7 Applications of MPC

- Distillation column
- Hydrocracker
- Pulp and paper plant
- Servo mechanism
- Robot arm ...

Chapter 3

Process Model

3.1 **MODELS**

Many different types of models are possible for calculating the predicted values of the process outputs, which are used in evaluating at discrete steps. For linear MPC applications, most algorithms use one of the three models namely state space model, step response model and impulse response model. It is a good practice to make use of discrete models for the output prediction whenever possible. Also step and impulse response models are very common in their use in MPC algorithms.

3.2 State Space Model

The state space representation is a time domain approach which provides a convenient and compact way for model representation and analysis of systems with multiple inputs and outputs.

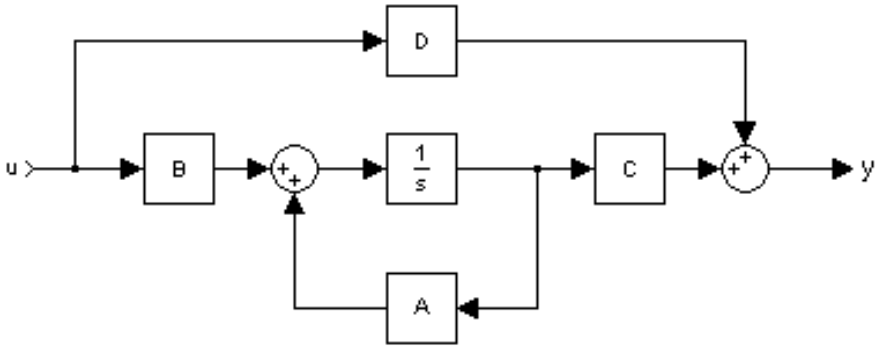

Fig: 3.1 Block diagram of State Space model (ref. [6])

A very general state space model of a linear system with *p* inputs, *q* outputs and *n* state variables can be written in the following form:

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t)$$
$$\mathbf{y}(t) = C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t)$$

where:

$\mathbf{x}(\cdot)$ is called the "state vector", $\mathbf{x}(t) \in \mathbb{R}^n$;

$\mathbf{y}(\cdot)$ is called the "output vector", $\mathbf{y}(t) \in \mathbb{R}^q$;

$\mathbf{u}(\cdot)$ is called the "input or control vector", $\mathbf{u}(t) \in \mathbb{R}^p$;

$A(\cdot)$ is the "state matrix", $\dim[A(\cdot)] = n \times n$,

$B(\cdot)$ is the "input matrix", $\dim[B(\cdot)] = n \times p$,

$C(\cdot)$ is the "output matrix", $\dim[C(\cdot)] = q \times n$,

$D(\cdot)$ is the " feed forward matrix", $\dim[D(\cdot)] = q \times p$,

$\dot{\mathbf{x}}(t) := \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t)$.

This is a generalized formula. Here all matrices are time-variant in nature (i.e., their elements can depend on time). However, in the common LTI case, matrices are time invariant. The time variable *t* can be continuous (e.g.$t \in \mathbb{R}$) or discrete (e.g.,$t \in \mathbb{Z}$). In the latter case, the time variable is usually denoted as *k*. Hybrid systems allow for time domains which have both discrete and continuous parts.

3.3 STEP RESPONSE MODEL

Step response models are obtained by making a unit step input change to a process operating at steady state. The model coefficients are same as the output values at each time step. Here “si” denotes the step response coefficients for the ith sample time after a unit step input change is made. If there is a non-unit step change, the output is scaled accordingly. This is also known as the convolution model.

The finite step response model is a vector of step response coefficients can be represented as

$$S=[\ s1\ s2\ s3\ s4\ s5\ \ldots\ sN]'$$

Where the model length N is long enough to ensure that the coefficients values are relatively constant.

3.4 IMPULSE RESPONSE MODEL

This is a very common form of model known as impulse response model. Here a unit pulse is applied to the manipulated input and the model coefficients are simply the values of the outputs the ith impulse response coefficients.

There is a direct relationship between step and impulse response models:

$Hi=Si-Si-1$
$Si=\Sigma\ hj$

The impulse response coefficients are simply the changes in the step response coefficients at each time step. Likewise, step response coefficients are the sum of the impulse response coefficients to that point. It should be noted that there are two major limitations in regards to step and impulse response models. They can only be used to represent open-loop stable processes and they require a large number of parameters or model coefficients as compared to state space and transfer function models.

Chapter 4

Cost Function

4.1 What is Cost Function

Cost function or objective function is a measure of performance of the process model as described in the previous chapter. It is required that the controlling system should follow a particular pattern. This is achieved by minimizing the cost function or objective function.

4.2 Types of Cost Function

There are several different choices for objectives functions. A few of them would be standard least-squares or quadratic objective function, absolute value objective function, etc.

4.3 Quadratic Objective Function:

This type of objective function is the sum of squares of the predicted errors and the control moves. Predicted errors are the differences between the set points and model-predicted outputs. Control moves are defined as the changes in control action from step to step.

A quadratic objective function for a prediction horizon of 2 and a control horizon of 1 can be written as

$$\Phi = (Rk+1 - Yk+1)^2 + ((Rk+2 - Yk+2)^2 + w\Delta Uk^2$$

In the above equation, Y represents the model predicted output, R is the set point, ΔU is the change in manipulated input from one sample to the next and w is a weight for the changes in the manipulated input. The subscripts indicate the sample time for which the objective function is being taken. k denotes the current sample time.

For a prediction horizon of P and a control horizon of M, the least Squares objective function can written as

$$\Phi = \Sigma (Rk+1 - Yk+1)^2 + w\Sigma \Delta Uk+1^2$$

4.4 Absolute Value Objective Function:

This is another possible objective function which simply takes a sum of the absolute values of the predicted errors and control moves. Though this type of objective function is fairly simple as compared to quadratic objective function, the later is more suitable in its approach handling the non linear process models.

For a prediction horizon of 2 and a control horizon of 1, the absolute value objective function is

$$\Phi = | (Rk+1 - Yk+1) | + | (Rk+2 - Yk+2) | + w| \Delta Uk|$$

Quite similar to quadratic objective function, it has the following general form for a prediction horizon of P and a control horizon of M:

$$\Phi = \Sigma | (Rk+1 - Yk+1)| + w\Sigma |\Delta Uk+1|$$

The optimization problem in hand is solved as a result of minimization of the objective function. This is obtained by adjusting the M control moves, subject to modeling equations and constraints on the inputs and outputs.

Hence, **Min Φ**

4.5 When to use a particular cost function

Minimization of objective functions by least-squares method is by far the most common objective function in MPC. Least squares method gives analytical solutions for unconstrained problems. It also compensates relatively larger errors more than smaller errors.

The absolute value objective function is used in a few algorithms because linear programming problem results obtained during optimization. Linear programming is frequently solved in large scale allocation and scheduling problems. For an example, an oil company often uses a linear programming to decide how to distribute oil to various refineries. Also linear programming is useful to decide how much and what product to produce at each plant. However, the linear programming approach is not useful for model predictive control. It is because the manipulated variable often moves from one extreme constraint to another.

Chapter 5

Optimization

5.1 Optimization Technique

Here optimization means solving the MPC algorithm. The optimization technique consists of minimizing the objective function defined in the problem statement. The solution from the minimization of the objective function gives adequate input signal that makes the system output follow the reference trajectory. The reference is chosen or set a priory and is referred to as X*ref*. Assuming that the reference trajectory is reachable and due to the system dynamics, each state reference has a corresponding input reference which is referred to as U*ref*.

5.2 Receding Horizon Control

Model Predictive Control is based on an iterative and finite horizon optimization of a plant model. At any time "t", the current plant state is sampled and a cost minimization algorithm is applied for a relatively short time horizon in the future: $[t, t + T]$. Only the first step of the control algorithm is implemented and then the plant state is sampled again. The calculations are again repeated starting from the now current state thereby yielding a new control and a new predicted state path. The prediction horizon keeps being shifted forward and for this reason model predictive control is also called **receding horizon control**. In practice it gives very good results although this approach is not optimal.

5.3 A schematic graph of receding horizon control

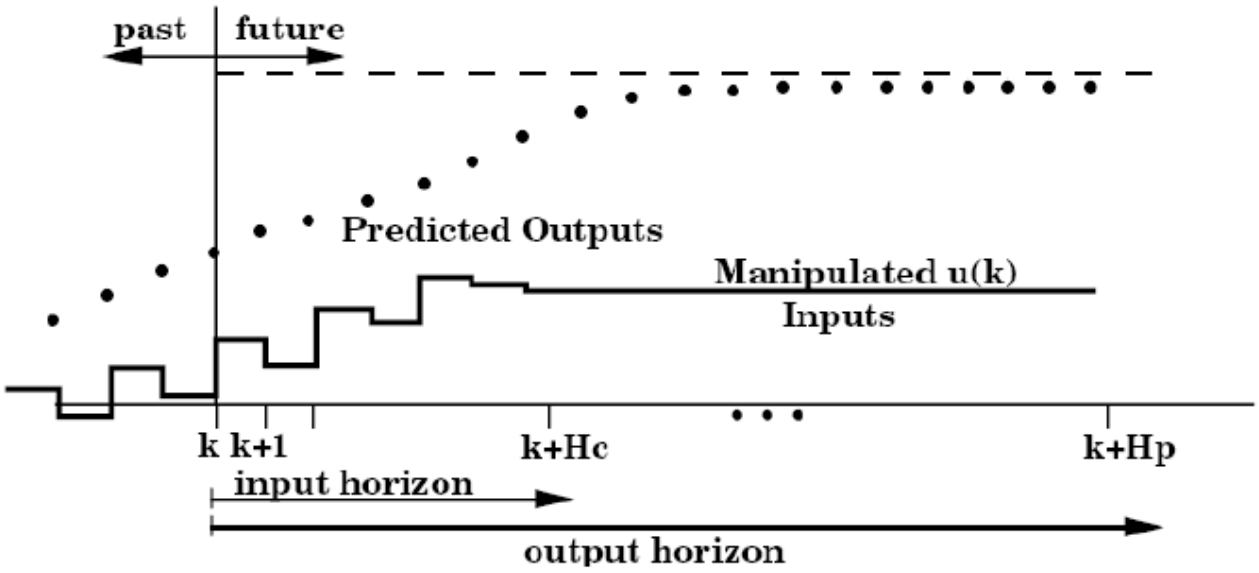

Fig. 5.1 Receding Horizon Control (ref. [3])

5.4 Explanation

In the above figure, we assume the control algorithm to be for a single-input, single-output (SISO) plant. Also we assume a discrete-time setting. The current time is labeled as time step k. The plant output at the current time is y(k). The figure shows the previous history of the output trajectory. A set point trajectory is shown which is the trajectory that the output should follow. The value of the set-point trajectory at any time t is denoted by s(t). Also a Reference Trajectory is shown which starts at the current output y(k). It defines an ideal trajectory along which the plant should return to the set-point trajectory, for example after a disturbance occurs. It is not necessary for the plant to go back to the set-point trajectory as fast as possible. Usually, the reference trajectory approaches the set point trajectory exponentially.

Let Tref denote defining the speed of response.

Now the current error is

$$€ (k) = s (k) - y(k)$$

Then the reference trajectory is chosen such that "i" steps later the error would be

$$€(k+i) = \exp(-iTs/Tref) * € (k)$$
$$= \lambda i * €(k)$$

where Ts is the sampling interval and $\lambda = \exp(-Ts/Tref)$.(note that 0<_<1). Now the reference trajectory is defined to be

$$r(k+i|k) = s(k+i) - €(k+i)$$
$$= s(k+i) - \exp(-Ti/Ts) * €(k)$$

The notation r(k+i|k) indicates that the reference trajectory depends on the conditions at time "k" in general.

Chapter 6

Dynamic Matrix Control

6.1 DYNAMIC MATRIX CONTROL:

Transfer function models are used to represent the dynamic behavior of a process. We normally use first order models with time delay. Transfer function models need the order to be specified. Here another way is to use a "discrete response model". It has the advantage that the model coefficients can be obtained directly from the state response which can be represented as:

$$X(k+1)=AX(k) + Bu(k) \quad (6.1.1)$$

$$Y(k+1)=CX(k) \quad (6.1.2)$$

Here X(k) : input at instant k.

U(k) : unit step function.

X(k+1),Y(k+1) : input and output at the next step.

A, B, C: state space coefficient of the model.

Let us consider control horizon "Nu" as no. of control actions that are taken in a model with prediction horizon of "Np".

Hence we have

- a=[a1,a2,a3,……,aN]' represent the state response coefficient vector.
- If the current time instant is k the control action has to be taken at time instant (k-1).

The predicted output of the process at instant k can be represented a by the equation:

$$y(k) = \sum_{i=1}^{N}(a_i)\,\Delta u(k-i) + a_{ss}u(k-N-1) + d(k) \qquad (6.1.3)$$

d(k) is the effect of disturbance.

So we can write the above equation as:

$$d(k) = y_{measured} - \sum_{i=1}^{N}(a_i)\,\Delta u(k-i) + a_{ss}u(k-N-1) \qquad (6.1.4)$$

for one step ahead prediction we can write equation 6.1.3 as

$$y(k+1) = a_1\,\Delta u(k) + a_2\Delta u(k-1) + \cdots + a_{ss}u(k-N+1) + d(k) \qquad (6.1.5)$$

since prediction horizon is greater than 1 , hence we can generalize equation 6.1.5 as:

$$\begin{bmatrix} Y(k+1) \\ Y(k+2) \\ \dots \\ \dots \\ \dots. \\ Y(k+ Np) \end{bmatrix} = \begin{bmatrix} a_{ss}u(k-N) \\ a_{ss}u(k-N+1) \\ \dots\dots\dots \\ \dots\dots\dots \\ \dots\dots\dots \\ a_{\dots}u(k-N+ Np+1) \end{bmatrix} + \begin{bmatrix} a_2 & a_3 \dots\dots a_N \\ a_3 & a_4 \dots\dots\dots 0 \\ \dots\dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots\dots \\ a_{\dots} & \dots\dots\dots\dots\dots 0 \end{bmatrix} \begin{bmatrix} \Delta u(k-1) \\ \Delta u(k-1) \\ \dots\dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots\dots \\ \Delta u(k+N-1) \end{bmatrix}$$

$$+ \begin{bmatrix} a_1 & 0 \dots\dots 0 \\ a_2 & a_1 \dots\dots\dots 0 \\ \dots\dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots\dots \\ a_{\dots} & \dots\dots\dots \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \dots\dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots\dots \\ \Delta u(k+N-1) \end{bmatrix} + \begin{bmatrix} d(k) \\ d(k) \\ d(k) \\ \dots\dots \\ \dots\dots \\ \dots\dots \\ d(k) \end{bmatrix}$$

Here we are assuming that disturbance d(k) is constant for all values of prediction variables.

Hence the above equation can be written as:

$$Y(k+N) = y_{past} + A\Delta U(k) + D$$

$$y_{past} = \begin{bmatrix} a_{ss}u(k-N) \\ a_{ss}u(k-N+1) \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ a_{\_\_}u(k-N+Np+1) \end{bmatrix} + \begin{bmatrix} a_1 & 0 \ldots\ldots 0 \\ a_2 & a_1 \ldots\ldots\ldots 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_{\_\_} \ldots\ldots\ldots \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \Delta u(k+N-1) \end{bmatrix}$$

Let us consider the desired output trajectory as:

$$Ysp = \begin{bmatrix} Ysp(k+1) \\ Ysp(k+2) \\ \ldots \\ \ldots \\ \ldots\ldots \\ \\ Ysp(k+Np) \end{bmatrix}$$

To get the perfect output we have Ysp(k+l) = Y(k+l)

Hence we have:

$$\Delta U = A^{-}(Ysp - y_{past} - D)$$

Hence our objective of finding the control moves can be achieved as above.

6.2 DMC Tuning

Every controller design has some design parameters, which can be tuned to get the desired response of the controller. These parameters are called the tuning parameters of the controller. The following guidelines are basically used to tune a DMC:

1 .The model horizon N should be selected so that $N\Delta t \geq$ open loop settling time. Values of N is normally taken between 20 to 70.

2. The prediction horizon Np determines how far into the future the control objective reaches.

Increasing Np makes the control more accurate but increases the computation.

The most recommended value of Np is when Np=N+ Nu.

3. The control horizon Nu determine the no of the control actions calculated into the future.

Too large value of Nu causes excessive control action.

Small value of Nu makes the controller insensitive of noise.

Chapter 7

Simulation Output in MATLAB

7.1 Output in MATLAB window

```
enter plant model in state space
enter A matrix [-2.5544 1.4354; 0.3235 -2.1245]
enter B matrix [7; -1.0242]
enter C matrix [0 2]
enter D matrix [0]

numm =

         0   -0.1653    0.1597


Sf =

         0
   -0.1653
   -0.2678
   -0.3271
   -0.3569
   -0.3671
   -0.3646
   -0.3542
   -0.3393
   -0.3222


u =  0     0     0     0     0     0     0     0     0     0
```

7.2 Output Figures

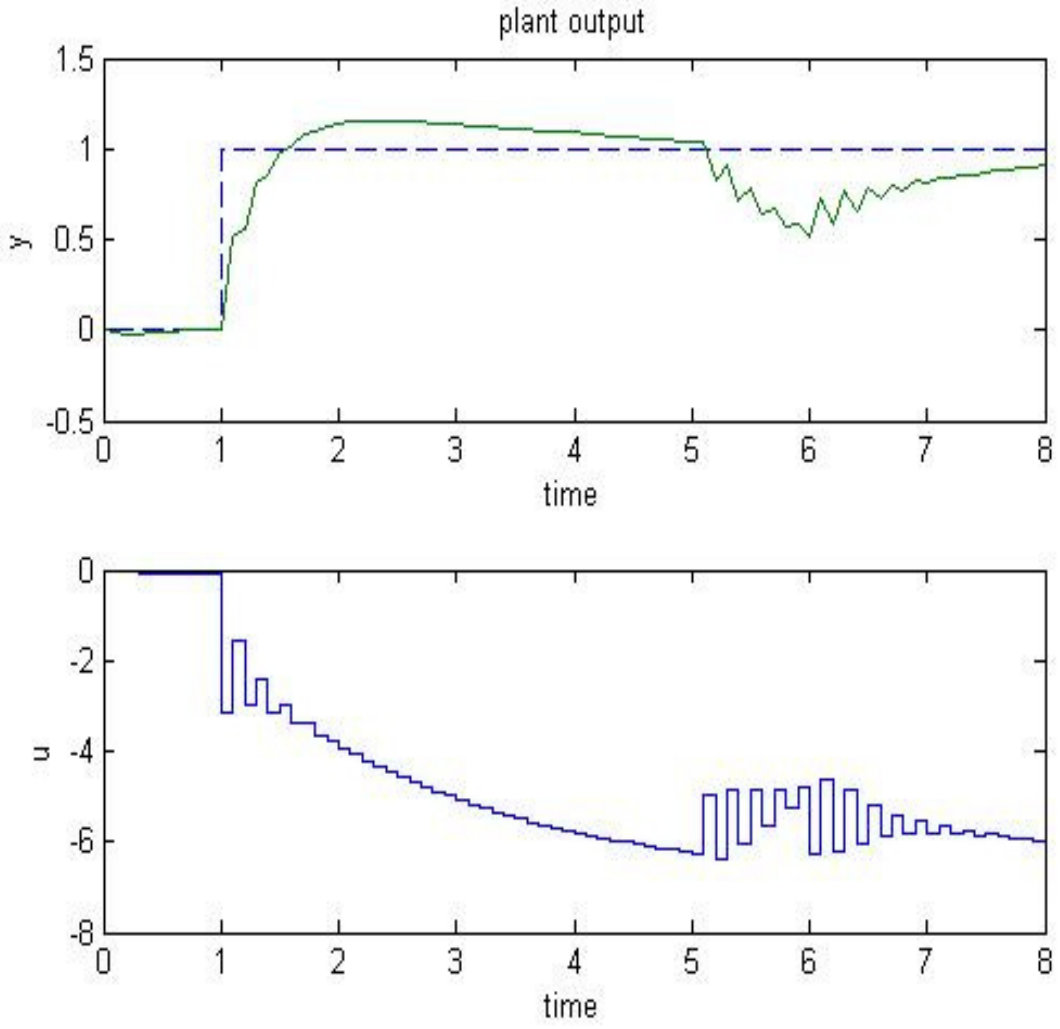

Fig 7.1 MPC output (for prediction horizon 10 and model length 50)

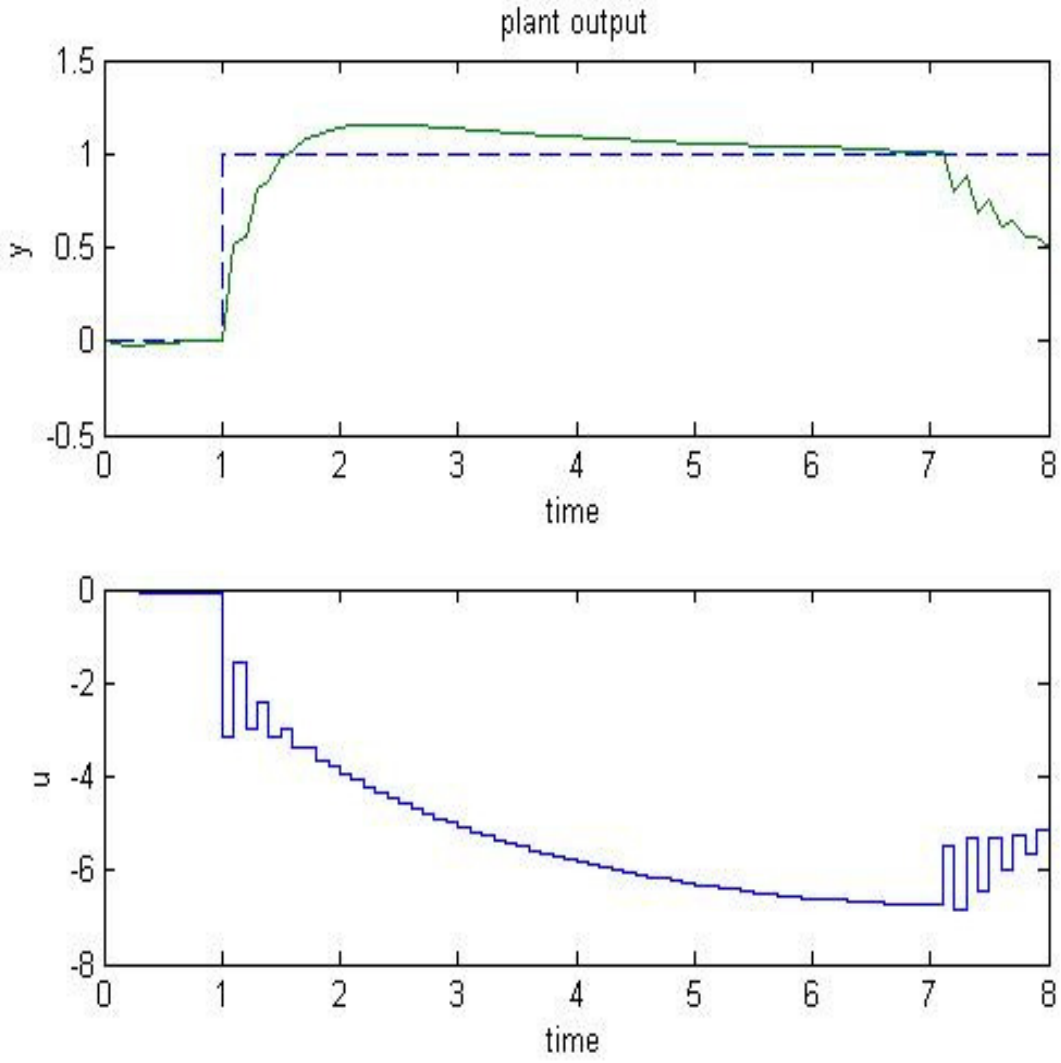

Fig 7.2 MPC output (for prediction horizon 10 and model length 70)

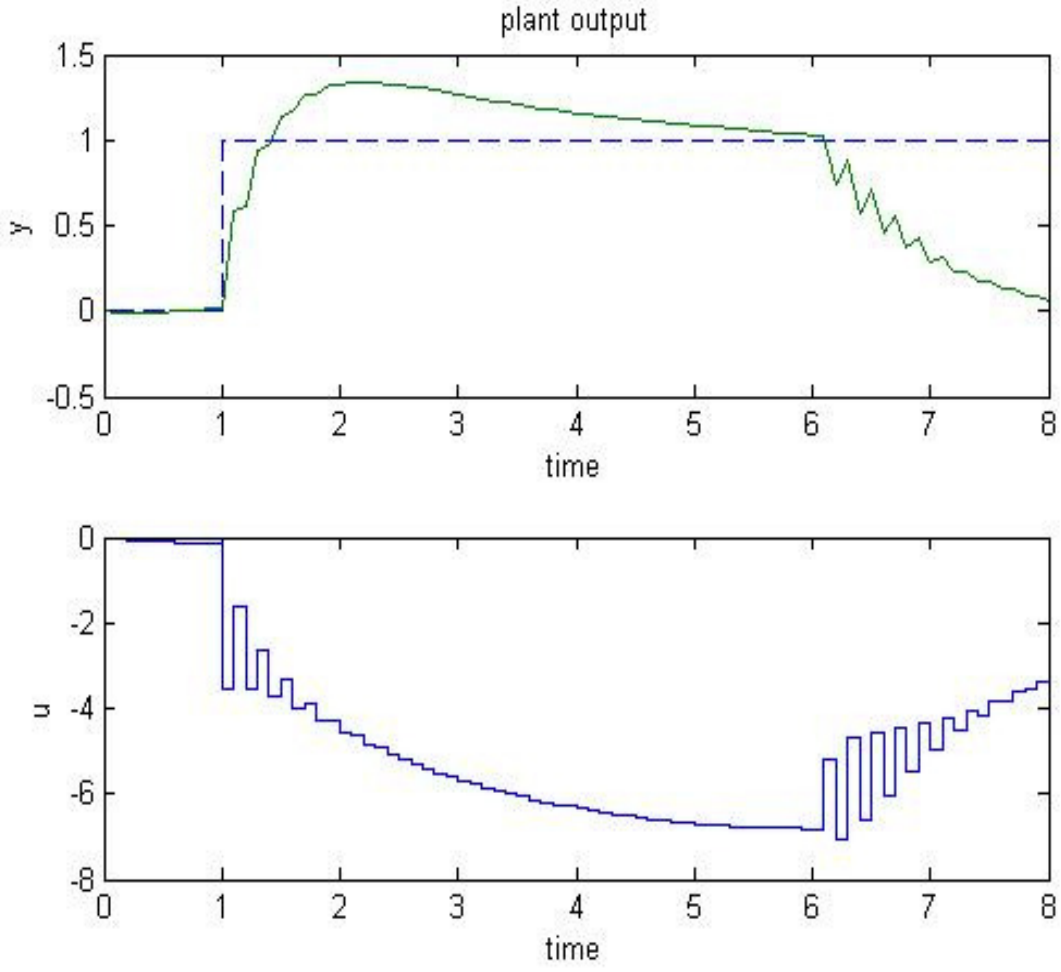

Fig 7.3 MPC output (for prediction horizon 20 and model length 70)

7.3 Inference

If we fix control horizon, then it is found out that taking a smaller prediction horizon results in the set point being achieved in much smaller time. However a shorter prediction horizon is more sensitive to the uncertainties in the model. Choosing a smaller model length does not capture the complete dynamics of the process. This results in a model error and poor performance.

Chapter 8

Conclusion

8.1 Conclusion

In this project, a linear dynamic matrix control algorithm is developed in the MATLAB. Simulations show that parameters like the model length, prediction horizon and the control horizon must be suitably chosen and calculated. That way, the control algorithm leads the system to the desired set point. Simulations show that there is no need for a very long prediction horizon. Feasibility and stability is assured even for short prediction horizons which guarantee that the output reaches the set point. Hence, short horizons are preferred in this project as the computational time is less. In addition to obtaining a control algorithm which works well, another objective is to make sure computational time and complexity of the optimization problem are reduced without losing performance and precision. In this project, various parameters of the DMC algorithm are set in such a way so as to suit our objective.

8.2 Future work

Due to time constraint, there are some interesting works which are still left to be done on this project. To be specific, it is worthwhile to test the controller on the real plant and find out how it behaves under the effect of noise or disturbances. Also further investigations can be done to reduce the computational time of the algorithm. A variety of cost functions or objective functions can be considered to reduce the computational time while maintaining good precision and performance.

References

1. Basics of Model Predictive Control , P.E. Orukpe
2. Nonlinear Model Predictive Control of the Four Tank Process, Ivana Drca
3. Introduction to process control by José Alberto Romagnoli, Ahmet Palazoğlu
4. Process Control Modeling design and simulation, B.W. Bequette, Prentice hall edition
5. www.mathworks.com
6. http://en.wikipedia.org